

REMARKS

Claims 1-20 are pending in the application, with claims 1, 9, and 12 being the independent claims.

Claims 1, 9, and 12 have been amended.

Applicant respectfully traverses the rejection of each independent and dependent claim in the application.

Specification

The present Action requires the withdrawal of amendments to the specification contained in the response filed August 31, 2007. As discussed with the Examiner on February 19, 2008, the change of “shared” to “cascading” is not new subject matter, but merely descriptive of matter already in the application. Support for use of “cascading” to describe the “execution code” may be found in the application in, for example, paragraphs 34-36 of the specification and in Figures 4, 5, and 6A-6G.

Applicant respectfully requests withdrawal of this requirement and entry of the amendments to the specification.

Rejections under 35 U.S.C. § 112

Claims 1-20 are rejected under 35 U.S.C. 112 for failing to comply with the written description requirement, for including subject matter not described in the specification. Applicant respectfully traverses these rejections. As discussed with the Examiner on February 19, 2007, the amendments made to claims 1-20 do not add new subject matter not already disclosed in the specification for the same reasons as stated above with regard to amendments to the specification. The change in phraseology from “shared execution code” to “cascading execution code” describes the already disclosed structure of the execution code and prevents overbroad interpretation of the term “shared.” The additional material added to the claims is supported in the specification, specifically paragraphs 34-36, and Figure 4, 5, and 6A-6G, as described above.

Therefore, claims 1-20 are allowable over the rejection for failing to comply with the written description requirement of 35 U.S.C. 112.

Rejections under 35 U.S.C. § 103

Claims 1-20 are rejected under 35 U.S.C. 103(a) as being obvious in light of the prior art in the field.

Claims 1-20 are rejected as being unpatentable over U.S. Patent No. 6,298,434 (hereinafter "Lindwer") in view of U.S. Patent No. 6,606,743 (hereinafter "Raz"), and in further view of U.S. Patent No. 5,734,908 (hereinafter "Chan"). Applicant respectfully traverses the rejection.

As discussed with the Examiner on February 19, 2008, Claims 1, 9 and 12 have been amended to clarify that the entry tier may be selected from any tier in the cascading execution code, and that execution will include all tiers below the entry tier in the cascading execution code. With respect to independent claims 1, 9 and 12, the combination of Lindwer, Raz and Chan does not teach or suggest the claimed embodiments. By Office's admission, Lindwer does not teach or suggest determining an entry point into shared execution code based on the stack state.

Raz does not supplement Lindwer to teach or suggest the claims. Raz does not supplement Lindwer to teach or suggest performing a stack-state-aware translation of the instruction to threaded code to determine an operand stack state for the instruction. The Action states that "and the code is threaded (Raz: column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention" (Action, pg. 4). As discussed with the Examiner on February 19, 2007, threaded code does not refer to the concept of multithreading, in which a single processor runs multiple programs at the same time by switching between them rapidly, but to the computer programming concept of threaded code. "Threaded code" is a term of art in computer programming for a technique for producing very compact code, which may be composed entirely of subroutine calls written as memory addresses containing the memory locations of the subroutine being called. The "cascading execution code" of the present claims is "threaded code," in that it is written using techniques for creating compact code, such as the example cited above of writing subroutine calls as memory addresses. "Threaded code" is an entirely distinct concept from multithreading, despite shared use of the word "thread." Raz discloses the use of multithreading, but does not teach or suggest the use of threaded code.

By Office's admission, Lindwer and Raz do not teach cascading execution code, wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier, each tier comprising at least one computer-executable instruction, and wherein the execution comprises entering the cascading code at a n entry tier indicated by the determined entry point and executing the entry tier at least one tier below the entry tier. Chan does not supplement Lindwer and Raz to overcome this deficiency.

Chan relates generally to software compilation, and specifically to a method for optimizing compiled software to be executed on a processor (Chan, col. 1, lines 35-55). The process of optimization in Chan first requires that code be transformed into an Intermediate Representation made up of a number of blocks (Chan, col. 2, lines 27-34). Optimization is performed on the Intermediate Representation by selecting pairs of blocks from the Intermediate Representation and moving instructions from one block to another block whenever possible and profitable in terms of execution time and resource usage of the compiled code (Chan, col. 3, lines 7-20). Chan does not disclose or suggest "cascading execution code." The blocks in Chan are never executed, in any order. The blocks of Chan, as stated above, form an Intermediate Representation. An Intermediate Representation is not executable. It is an intermediate state for code that is in the process of being compiled into code that is executable. Chan refers to blocks "executing", only in terms of the order in which the instructions contained in the blocks will execute once the blocks have been compiled into an executable code.

Chan does not disclose wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier, wherein an entry tier is selected from any tier of the cascading execution code and the entry tier and all tiers below the entry tier are executed. A basic block is merely a collection of instructions within an Intermediate Representation. The basic blocks that make up an Intermediate Representation in Chan are not enterable, as they are not executable, and they are not tiered, as it is possible for the basic blocks to be structured in a circular manner. Tiers cannot be circular, as tiers require that when a second tier is below a first tier, that second tier must not be above any tier that is above the first tier. The basic blocks of Chan may be arranged in a circular fashion in which a first block is above a second block and below a third block, while the second block is below the first block and above the third block. Blocks in Chan do not

adhere to the tiered structure of cascading execution code, and further, are not enterable at any tier, as they are abstract collections of instructions which are not in themselves executable.

Therefore, claims 1, 9, and 12 are allowable over Lindwer in view of Raz in further view of Chan.

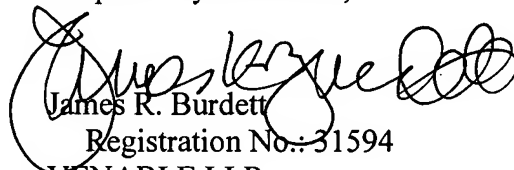
Claims 2-8, 10, 11, and 13-20 are allowable for at least for depending from the independent claims which are allowable over Lindwer and Raz in view of Chan, as discussed above.

Conclusion

All of the stated grounds of rejection have been properly traversed. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding rejections and that they be withdrawn. Applicants believe that a full and complete reply has been made to the outstanding Office Action and, as such, the present application is in condition for allowance. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is hereby invited to telephone the undersigned at the number provided.

Dated: February 28, 2008

Respectfully submitted,



James R. Burdett

Registration No.: 31594

VENABLE LLP

P.O. Box 34385

Washington, DC 20043-9998

(202) 344-4000

(202) 344-8300 (Fax)

Attorney/Agent For Applicant